I Présentation de la récursivité

La récursivité est une technique de programmation où une fonction s'appelle elle-même pour résoudre un problème.

Voici des implémentations respectivement itérative et récursive de la fonction factorielle :

```
1 def factorielle_iterative(n):
2     resultat = 1
3     for i in range(1, n + 1):
4         resultat *= i
5     return resultat
```

```
1 def factorielle_recursive(n):
2     if n == 0:
3         return 1
4     else:
5         return n * factorielle_recursive(n - 1)
```

II Éléments d'une bonne programmation récursive

Pour qu'une fonction soit correctement conçue de manière récursive, il est nécessaire que trois éléments soient parfaitement identifiés et impléentés

1. **Condition d'arrêt** : Il est crucial d'avoir une condition qui termine la récursion (typiquement : une grandeur entière qui atteint sa valeur minimale : 0,1, ou 2).
2. **Appel récursif** : La fonction doit s'appeler elle-même avec une grandeur modifiée, généralement en une valeur strictement inférieure, pour se rapprocher de la condition d'arrêt.
3. **Gestion des cas de base** : Il faut gérer correctement les cas de base pour éviter des appels récursifs infinis.

III Exercices de programmation récursive

■ Exercice 1.

Écrire une fonction `suite_arithmetique` qui calcule le n-ième terme d'une suite arithmétique où chaque terme est défini par la relation :

$$a_n = a_{n-1} + d$$

où a_0 est le premier terme et d est la différence.

Arguments d'entrée : un entier n, un entier a_0, un entier d

Sortie : le n-ième terme a_n.

M. Patel
2024-2025
Classes de B_2, Rennes

■ **Exercice 2.**
Écrire une fonction nommée `suite_geometrique_recursive` qui calcule le n-ième terme d'une suite géométrique, définie par la relation :

$$u_n = r \cdot u_{n-1}$$

où u_0 est le premier terme et r est la raison de la suite.
Arguments d'entrée :
- Un entier n (l'indice du terme à calculer).
- Un flottant u_0 (le premier terme de la suite).
- Un entier r (la raison de la suite).

Sortie : Le n-ième terme $g(n)$ de la suite géométrique.

■ **Exercice 3.**
Écrire une fonction `somme_suite` qui calcule la somme des premiers termes d'une suite définie par :

$$S_n = \begin{cases} 0 & \text{si } n = 0 \\ S_{n-1} + n & \text{sinon} \end{cases}$$

Arguments d'entrée : un entier n
Sortie : la somme S_n.

■ **Exercice 4.**
Écrire une fonction `fibonacci` qui calcule le n-ième terme de la suite de Fibonacci :

$$F_n = \begin{cases} 0 & \text{si } n = 0 \\ 1 & \text{si } n = 1 \\ F_{n-1} + F_{n-2} & \text{sinon} \end{cases}$$

Arguments d'entrée : un entier n
Sortie : le n-ième terme F_n.

■ **Exercice 5.** Utiliser la fonction pour calculer la valeur de F_{18}. Que constate-t-on ?

■ **Exercice 6.** Écrire une fonction `liste_consec` qui prend en entrée un entier n et renvoie en sortie la liste des entiers de 1 à n.
Arguments d'entrée : un entier.
Sortie : une liste d'entiers.

■ **Exercice 7.** Écrire une fonction `somme_liste` qui prend en entrée une liste de flottants et renvoie leur somme.
Arguments d'entrée : une liste de flottants
Sortie : la somme des éléments de la liste.

■ **Exercice 8.**
Écrire une fonction `produit_liste` qui prend en entrée une liste de flottants et renvoie le produit de ses éléments.
Arguments d'entrée : une liste de flottants
Sortie : le produit des éléments de la liste.

■ **Exercice 9.**
Écrire une fonction `inverser_chaine` qui inverse une chaîne de caractères.
Arguments d'entrée : une chaîne de caractères
Sortie : la chaîne inversée.

TP 5 : récursivité

■ **Exercice 10.**
Écrire une fonction `compte_caracteres` qui prend en entrée une chaîne de caractères et renvoie le nombre de caractères dans cette chaîne.
Arguments d'entrée : une chaîne de caractères
Sortie : le nombre de caractères dans la chaîne.

■ **Exercice 11.** Écrire une fonction nommée `est_palindrome` qui vérifie si une chaîne de caractères est un palindrome.
Arguments d'entrée : une chaîne de caractères
Sortie : `True` si la chaîne est un palindrome, sinon `False`.

■ **Exercice 12.** Écrire une fonction nommée `max_liste` qui trouve le maximum d'une liste d'entiers.
Arguments d'entrée : une liste d'entiers
Sortie : le maximum de la liste.

M. Patel
2024-2025
Classes de B_2, Rennes